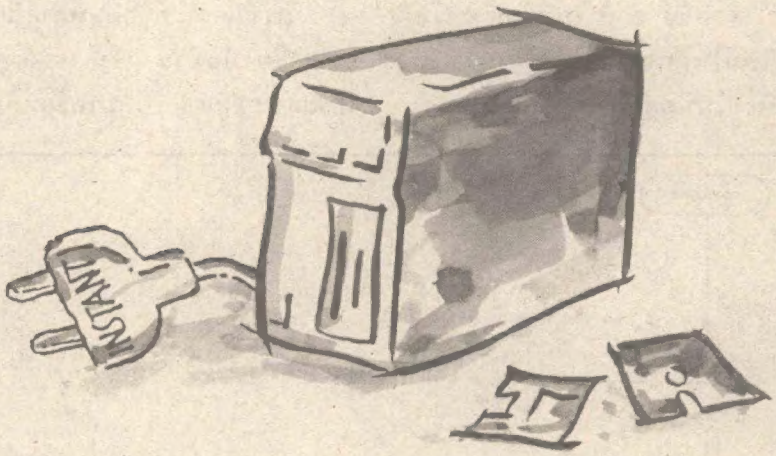
สมชาย นำประเสริญชัย

สารพัน UNIX
: เรื่องของโปรเซส

กล่าวถึงโปรเซสในเบื้องต้น เพื่อให้รู้จักและเห็นภาพในการทำงานของยูนิกซ์



คงมีผู้อ่านอีกจำนวนไม่น้อยที่ยังสงสัยอยู่ว่า UNIX หรือระบบคอมพิวเตอร์ที่เรียกกันว่ามัลติโปรแกรมมิง ทำไมจึงสามารถทำงานได้หลาย ๆ งานพร้อมกันไปไม่ว่างานต่าง ๆ นั้นจะเหมือนกัน หรือแตกต่างกันก็ตาม เช่น ในขณะที่ผู้ใช้กำลังใช้งานอยู่บนระบบนั้นปรากฏว่าเครื่องพิมพ์ของระบบก็ยังพิมพ์งานได้ต่อไปเรื่อย ๆ โดยไม่สนใจอะไร แถมไฟที่ตัวเครื่องยังกะพริบแสดงถึงการติดต่อดิสค์ในขณะนั้นอีกด้วย ในสารพัน.....UNIX ตอนนี้จะกล่าวถึงเรื่องราวของโปรเซสอย่างง่าย ๆ เพื่อพอเป็นแนวทางในการไขปัญหา แต่สิ่งที่จะต้องเข้าใจคือ โปรเซสนั้นคืออะไร มีขั้นตอนหรือหลักการทำงานอย่างไร

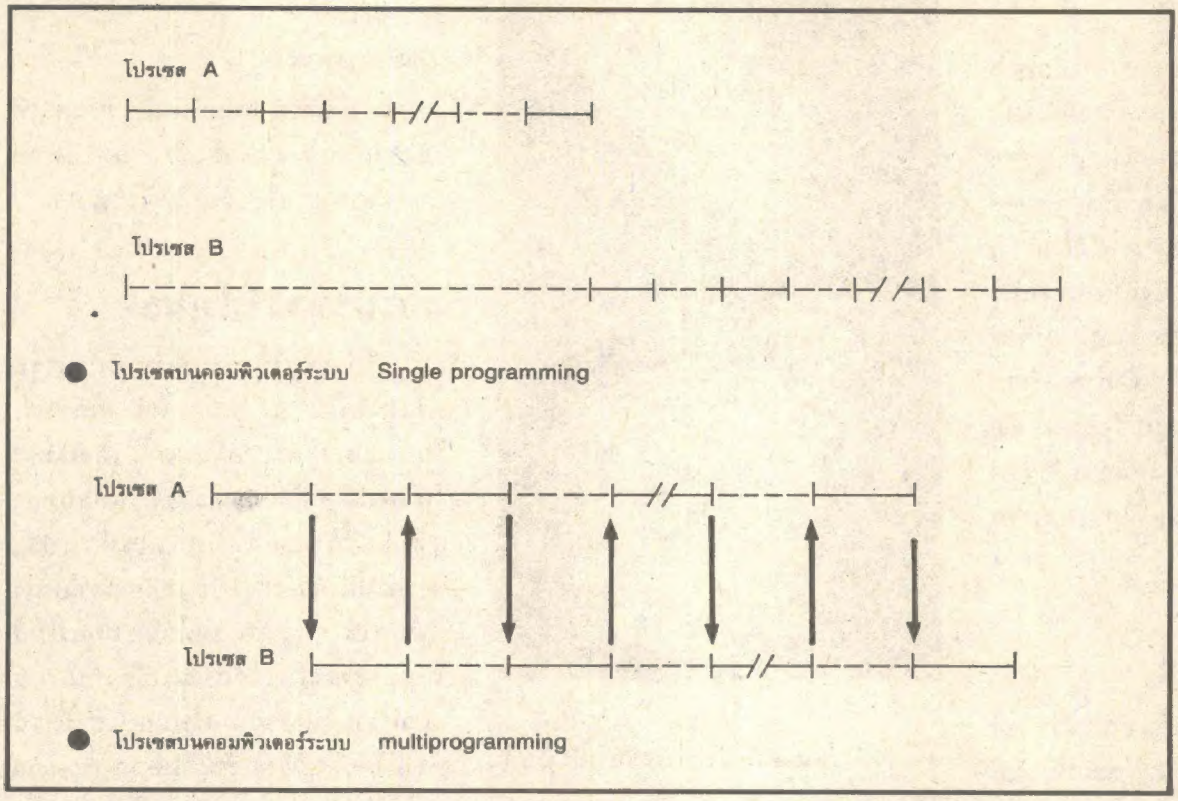
โปรเซสคืออะไร

คำจำกัดความง่าย ๆ ของโปรเซสก็คือ โปรแกรมที่กำลังทำงานหรือกำลังรันอยู่นั่นเอง ในระบบยูนิกซ์และคอมพิวเตอร์ที่มีลักษณะเป็นมัลติโปรแกรมมิงในส่วนของตัวดำเนินการประมวลผลหรือที่เรียกกันว่าซีพียูนั้น จะมีความสามารถในการนำโปรแกรมต่าง ๆ ให้ทำงานสลับกันไปมาได้ โดยให้โปรแกรมหนึ่ง ๆ ทำงานในช่วงเวลาที่กำหนดโดยลำดับความสำคัญของโปรแกรมนั้น ๆ ในช่วงเวลาที่เท่ากันเมื่อเทียบกับการดำเนินงานแบบให้ทำงานเพียงโปรแกรมเดียว (Single programming) จะเห็นได้ว่าได้ประโยชน์การนำช่วงของการรอหรือติดต่อกับอุปกรณ์ต่าง ๆ มาใช้งาน และจากช่วงเวลาของโปรแกรมที่สลับกันไปมาอย่างรวดเร็วนี้เอง ทำให้ผู้ใช้มองเห็นว่างานหรือโปรแกรมต่าง ๆ นั้นได้ทำงานขนานกันไป (psudo parallelism) โดยความเป็นจริงแล้วจะมีเพียงโปรแกรมเดียวเท่านั้นที่ทำงานในเวลาหนึ่ง ๆ เพราะเป็นระบบซีพียูเดียว ส่วนการที่จะทำให้การทำงานต่าง ๆ ขนานกันไปจริงจะต้องมีการเพิ่มซีพียูเพื่อช่วยในการประมวลผล

โปรเซสประกอบด้วยตำแหน่งเนื้อที่ของหน่วยความจำที่เคอร์เนลจะต้องทำการจับจองเพื่อให้โปรเซสใช้, ส่วนของโปรแกรม และข้อมูลพร้อมทั้งรายละเอียดต่าง ๆ ที่จำเป็นต่อการดำเนินการของโปรเซส เช่น เจ้าของ, สถานะปัจจุบันของโปรเซส, หมายเลขโปรเซส, ลำดับความสำคัญ และข้อสรุปของการใช้ทรัพยากร

การดำเนินการกับโปรเซสในระบบยูนิกซ์

การสร้างโปรเซส โปรเซสถูกสร้างขึ้นด้วยโปรเซสเองยกเว้นโปรเซส 0 ถูกสร้างเมื่อเริ่มต้นระบบ โปรเซสจะเริ่มต้นโดยการ fork เพื่อให้ได้โปรเซสใหม่ขึ้น โดยถือว่าโปรเซสเดิมที่ใช้สร้างเป็นโปรเซสแม่ (parent process) และโปรเซสที่ถูกสร้างขึ้นใหม่ถือเป็นโปรเซสลูก (chile process) ซึ่งลักษณะของโปรเซสลูกจะเหมือนกับโปรเซสแม่จะต่างกันเพียง PID (Process IDentifi-



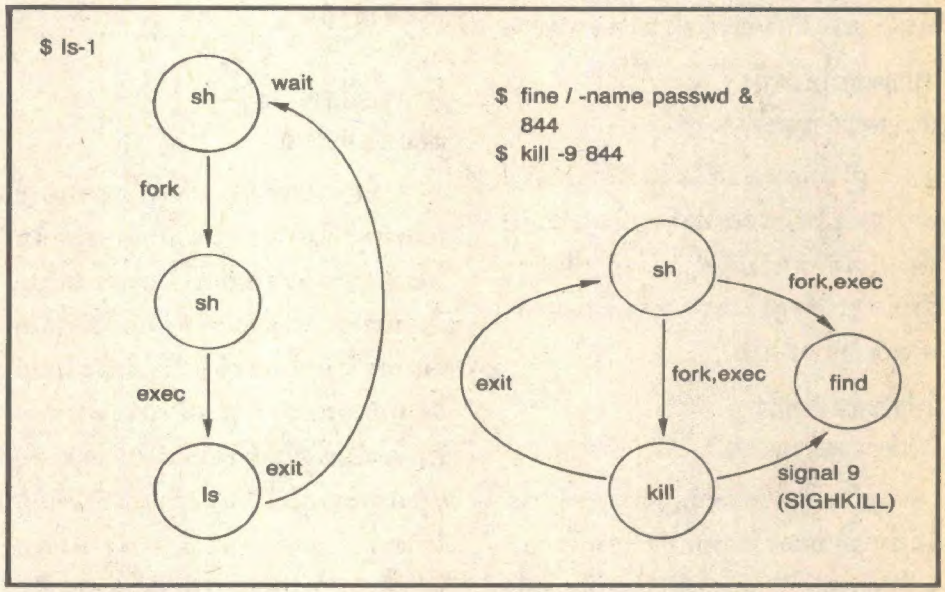
รูปที่ 1
แสดงการเปรียบเทียบของโปรเซสบนระบบคอมพิวเตอร์ที่ต่างกัน

cation), PPID (Parent Process IDentification) โดย PPID นี้จะเหมือนกับ PID ของโปรเซสที่สร้างมัน และโปรเซสใหม่จะถูกสันนิษฐานว่าไม่มีการใช้ทรัพยากรใด ๆ กลไกในการสร้างโปรเซสจะอาศัยคำสั่งสำคัญ ๆ 4 คำสั่ง คือ fork, exec, wait และ exit

- fork โปรเซสจะทำการก๊อบปี้ตัวเองเพื่อทำงานอย่างอื่นให้กับโปรเซสหลัก

- exec เป็นคำสั่งให้ UNIX ดึงโปรแกรมจากในดิสค์มาทับโปรเซสที่เรียกคำสั่งนี้ การที่จะใช้คำสั่งนี้เมื่อต้องการรันคำสั่งใด ๆ

- wait เป็นคำสั่งให้โปรเซสแม่หยุดการทำงานเพื่อรอโปรเซสลูกจบการทำงาน และจะรับค่าสถานะการทำงานครั้งสุดท้ายของโปรเซสลูกคืนมา



รูปที่ 2 การสร้างและการยกเลิกโปรเซส

- exit เป็นคำสั่งหยุดการทำงานของโปรเซส

การทำลายโปรเซส ทำได้โดยโปรเซสเรียกฟังก์ชัน exit หรือมีการส่งสัญญาณควบคุม เช่น คำสั่ง kill (SIGTERM), logout จากระบบ (SIGHUP) หรือเกิดจากความผิดพลาดของระบบ (Coredump)

การเลือกโปรเซส

ในสภาวะที่มีหลาย ๆ โปรเซสมีความต้องการที่จะรันบนซีพียูตัวเดียวกัน ส่วนของระบบจะต้องสามารถตัดสินใจได้ว่าโปรเซสใดจะเป็นโปรเซสที่จะรันถัดไป ส่วนของการตัดสินใจนี้เรียกว่า scheduling ซึ่งเป็นส่วนหนึ่งของฟังก์ชันหลักของเคอร์เนลในยูนิกซ์ โดยหลักการเลือกนั้นจะดูถึงลำดับความสำคัญของแต่ละโปรเซสซึ่งขึ้นอยู่กับเจ้าของโปรเซสนั้น ๆ และมีส่วนที่เรียกว่า dispatcher เป็นตัวโอนการควบคุมซีพียู

ชนิดของโปรเซส

ลักษณะของโปรเซสในระบบยูนิกซ์มีหลายแบบ ซึ่งแต่ละแบบจะมีความเหมาะสมในการทำงานที่แตกต่างกันไปตามลักษณะของโปรเซสนั้น ๆ

โปรเซสเบื้องหน้า
(forground process)

เป็นลักษณะของโปรแกรมหรือคำสั่งทั่วไปที่ขณะทำงานนั้นมีความต้องการติดต่อกับผู้ใช้ ไม่ว่าจะเป็นการรับข้อมูลเข้าสู่ระบบ หรือการแสดงผลลัพธ์ออกก็ตาม

โปรเซสฉากหลัง
(background process)

เป็นโปรเซสที่มีลักษณะการทำงานในฉากหลักและโปรเซสประเภทนี้ระบบจะให้ความสำคัญต่ำกว่าโปรเซสเบื้องหน้า ส่วนการกำหนดว่าควรให้โปรเซสใดทำงานในลักษณะของโปรเซสฉากหลังโดยโปรเซสฉากหลังจะใช้เวลาในการทำงานค่อนข้างนาน และไม่มีความจำเป็นที่จะต้องติดต่อกับผู้ใช้ เช่น การคอมพายล์โปรแกรม การที่จะให้คำสั่ง

หรือโปรเซสใดทำงานในฉากหลังนั้นก็เพียงเติมเครื่องหมาย anpersand (&) ท้ายคำสั่งนั้น ๆ

โปรเซสปีศาจ
(deamon trocess)

ชื่อนี้ขออนุญาตอาจารย์ภุชงค์มาใช้นะครับ โปรเซสชนิดนี้มีลักษณะเป็นโปรเซสฉากหลังโดยจะเริ่มทำงานตั้งแต่เริ่มต้นระบบจนถึงยกเลิกระบบ หลาย ๆ คนมองว่าโปรเซสชนิดนี้ไม่มีประโยชน์ใด ๆ เลยแต่จริง ๆ แล้วบรรดาเหล่าโปรเซสปีศาจนี้แม้เวลาส่วนใหญ่จะหลับอยู่ แต่จะถูกปลุกขึ้นมาทำงานทันทีเมื่อได้รับการกระตุ้นจาก I/O ซึ่งโปรเซสชนิดอื่น ไม่สามารถทำได้เพราะเป็นลักษณะการใช้ร่วมกันทำให้ไม่สามารถตอบสนองทันทีทันใด ลักษณะเด่นอีกอย่างหนึ่งของโปรเซสปีศาจก็คือ ไม่มี stdin และ stdout และยังมีหลาย ๆ คนกล่าวไว้ว่าโปรเซสที่มีชื่อลงท้ายด้วย d มักจะเป็นโปรเซสปีศาจด้วย

โปรเซสผีดิบ
(Zombie process)

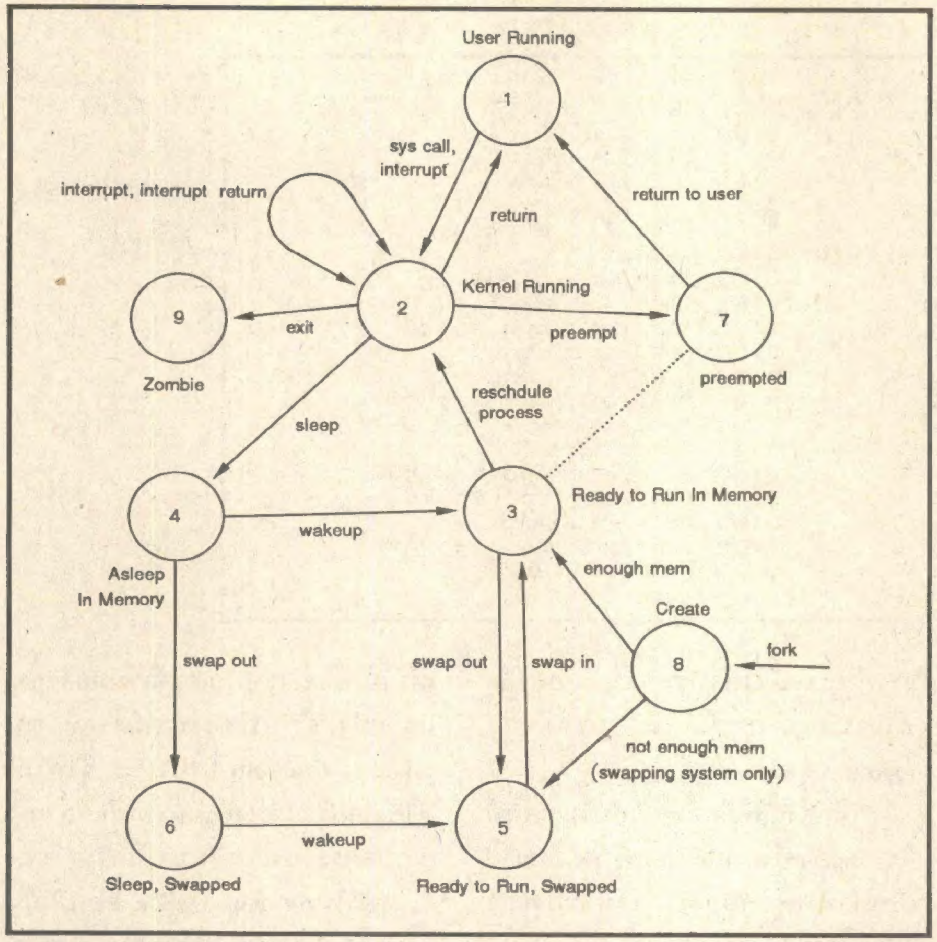
คำนี้ก็ยืมมาเช่นกัน เป็นโปรเซสที่ไม่สามารถทำงานได้แล้ว แต่ไม่หลุดออกจากตารางโปรเซสได้ด้วยตัวเอง

สถานะของโปรเซส

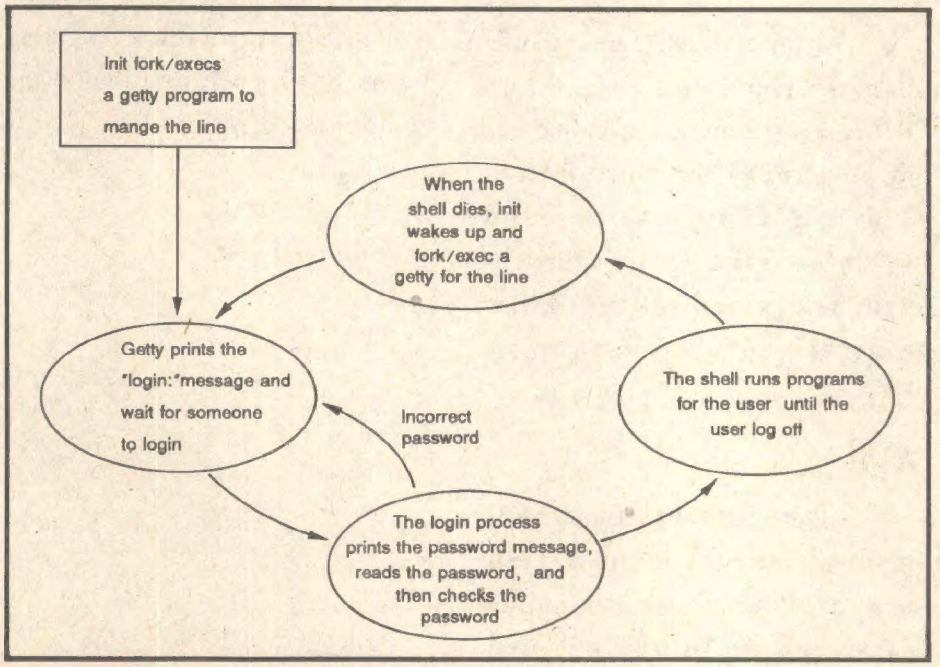
โปรเซสในระบบยูนิกซ์สามารถทำงานได้ใน 2 โหมด คือ ยูสเซอร์-โหมดและเคอร์เนลโหมด โดยแต่ละโหมดจะมีสแต็กของตนเอง ในส่วนของยูสเซอร์โหมดจะมีการใช้ตัวแปร, การผ่านค่าตัวแปร, ข้อมูล และตำแหน่งที่เรียกใช้ ส่วนในสแต็กของเคอร์เนลโหมดจะทำงานในลักษณะเดียวกัน การเข้าทำงานในเคอร์เนลโหมด สามารถทำได้โดยการเรียกซิสเต็มคอล (system call) เช่น write, open หรืออื่น ๆ สแต็กของเคอร์เนลนี้จะถูกใช้งานเมื่อ system call มีการเรียกซับรูทีนภายในเคอร์เนล

สถานะพื้นฐานของโปรเซสมีอยู่ 5 สถานะ ดังรูปที่ 3 ที่ควรทราบคือ

Running	เป็นสถานะที่โปรเซสกำลังรันอยู่
Sleeping	เป็นสถานะที่โปรเซสรอคอยอุปกรณ์อินพุต/เอาต์พุตบางอย่าง
Swapped	เป็นสถานะที่โปรเซสยังไม่ได้อยู่ในหน่วยความจำ
Zombie	เป็นสถานะที่โปรเซสพยายามยกเลิกตัวเอง หรือพยายามจะตาย
Ready	เป็นสถานะที่โปรเซสพร้อมที่จะรันได้



รูปที่ 3 แสดงสถานะต่าง ๆ ของโปรเซส



รูปที่ 4 สรุปขั้นตอนโปรเซสในขณะ login เป็นวงจรเหตุการณ์ที่เกิดขึ้นในแต่ละสายสื่อ

ขั้นตอนของโปรเซสที่เกิดขึ้นขณะ login

เมื่อเริ่มต้นระบบโปรเซส init ซึ่งเป็นโปรเซสหมายเลข 1 จะเริ่มทำงานโดยการ fork getty เพื่อให้รอรับชื่อผู้ใช้งาน หลังจากนั้นก็จะทำการ fork โปรเซส login เพื่อพิมพ์ password ออกมาเพื่อให้ผู้ใช้ป้อนรหัสผ่าน ต่อจากนี้ก็จะทำการตรวจสอบชื่อ, รหัสผ่านกับไฟล์ etc/passwd ว่าถูกต้องหรือไม่ ถ้าไม่ถูกต้องโปรเซส getty จะพิมพ์ login เพื่อให้ผู้ใช้เข้าสู่ระบบใหม่อีก หากตรวจสอบแล้วว่าชื่อและรหัสผ่านตรงตามไฟล์ etc/passwd แล้ว จึงเริ่มเข้าสู่ระบบโดยที่โปรแกรมแรกคือ เชลล์ (/bin/sh) ซึ่งการดำเนินการของเชลล์จะเป็นไปเรื่อย ๆ จนกระทั่งเชลล์ตายหรือมีการ logout ตัวโปรเซส init ก็จะไปปลุกโปรเซส getty เพื่อให้พิมพ์ login เพื่อรับการเข้าสู่ระบบของผู้ใช้ต่อไป

คำสั่งในการดำเนินการกับโปรเซส

ในกรณีที่เราสั่งให้โปรแกรมทำงานในฉากหลัง แล้วต้องการยกเลิกโปรแกรมนั้นจะทำอย่างไร สิ่งแรกที่จะต้องรู้คือหมายเลขประจำโปรเซส (PID) ซึ่งสามารถกระทำได้โดยใช้คำสั่ง ps (process status)

ps เป็นคำสั่งที่แสดงรายละเอียดว่าโปรเซสใดกำลังทำงานอยู่บ้าง ซึ่งรวมไปถึงบรรดาเหล่าโปรเซสปีศาจด้วย โดยจะบอกถึงหมายเลขโปรเซส, หมายเลขโปรเซสแม่ และอื่น ๆ หลังจากที่เราทราบหมายเลขโปรเซสแล้วก็สามารถให้

```
  UID   PID  PPID   C    STIME    TTY  TIME CMD
 root     1     0   0 17:06:01      -  0:00 /etc/init
 root  1961     1   0 17:07:43      -  0:00 /etc/syncd 60
 root  2245     1   0 17:07:48      -  0:00 /etc/srcmstr
 root  3053  2245   0 17:07:59      -  0:00 /usr/etc/nfsd 4
 root  3243     1   0 17:07:43      -  0:00 /usr/lib/errdemon
  snp  3341  4599  10 17:11:55  hft/0  0:00 ps -ef
 root  4328  7398   0 17:07:58      -  0:00 /usr/etc/biod 4
  snp  4599     1   1 17:08:01  hft/0  0:02 -ksh
 root  5108  2245   0 17:07:59      -  0:00 /usr/etc/rpc.statd
 root  5838  2245   0 17:07:52      -  0:00 /etc/syslogd
 root  6097  2245   0 17:07:53      -  0:00 /usr/lib/sendmail -bd -q30m
 root  6356  2245   0 17:07:54      -  0:00 /usr/etc/portmap
 root  6615  2245   0 17:07:54      -  0:00 /etc/inetd
 root  6874  2245   0 17:07:55      -  0:00 /etc/rwhod
 root  7922  2245   0 17:07:59      -  0:00 /usr/etc/rpc.mountd
 root  8182  2245   0 17:08:00      -  0:00 /usr/etc/rpc.lockd
 root  8697     1   0 17:08:01      -  0:00 /etc/cron
 root  9470  2245   0 17:08:08      -  0:00 /etc/qdaemon
 root  9725  2245   0 17:08:01      -  0:00 /etc/writesrv
```

รูปที่ 5
แสดงผลการทำงานของคำสั่ง ps-ef

โปรแกรมหรือโปรเซสนั้นหยุดทำงานได้โดยใช้คำสั่ง kill (คำสั่งฆ่า) เช่น

```
$ kill 373
```

จะเป็นการฆ่าหรือยกเลิกโปรเซสหมายเลข 373 แต่ในกรณีที่ต้องการยกเลิกหลายโปรเซส หรือต้องการฆาตกรรมหมู่ก็เพียงเติมหมายเลขของโปรเซสเข้าไปอีก เช่น

```
$ kill 374 375 377 385
```

แต่บางครั้งบางโปรเซสมีความอยู่ยงคงกระพันไม่ยอมตายง่าย ๆ ก็ต้องใช้วิธีการฆ่าอย่างรุนแรง โดยใช้คำสั่ง

```
$ kill -9 377
```

-9 เป็นการบอกระบบให้ระดับของสัญญาณฆ่านั้นมีความรุนแรงยิ่งขึ้น ถ้ายังไม่ตายอีก อาจเกิดความผิดปกติที่ระบบก็ได้

ในบางกรณีเราอยากทำงานทิ้งไว้ ถึงแม้ว่าเราจะ logout ออกจากระบบแล้ว ให้ใช้คำสั่ง nohup เช่น

```
$ nohup sortfile data1
```

จะเป็นการสั่งให้โปรแกรม sortfile ทำงานในฉากหลัง ถึงแม้จะมีการ logout ไปแล้ว

อีกคำสั่งหนึ่งที่เกี่ยวกับโปรเซสก็คือ nice หรือคำสั่งสุภาพบุรุษ เพราะเป็นคำสั่งเพื่อลดเวลาการทำงานของโปรแกรม (ทำงานช้าลง) เช่น

```
$ nice -10 sortfile data2
```

โปรแกรม sortfile จะถูกเพิ่มค่าประจำตัว เรียกว่า nice value ขั้น 10 ทำให้ทำงานช้าลงแนวความคิดของ nice คือ กรณีที่มีผู้ใช้จำนวนมากที่ทำงานไม่สำคัญให้ใช้ซีพียูน้อยลง เพื่อระบบจะได้ทำงานเร็วขึ้น ซึ่งการขัดกับธรรมชาติของมนุษย์ที่แก่งแย่งของดีเสมอ คำสั่งนี้จึงได้ชื่อว่าเป็นคำสั่งสุภาพบุรุษ

สรุป

โปรเซสเป็นสิ่งที่จะพบเห็นทั่วไปในระบบโอเอส แต่ในระบบเก่า ๆ เช่น ดอส การจัดการโปรเซสมักจะถูกมองข้ามความสำคัญไปเพราะอย่างไรก็สามารถทำงานได้เพียงโปรเซสเดียวเท่านั้น ซึ่งต่างจากโอเอสที่เป็นมัลติโปรแกรมมิง ซึ่งโปรเซสจะสามารถทำงานพร้อม ๆ กันได้หลายโปรเซส จึงทำให้ผู้ใช้งานเริ่มมองเห็นความสำคัญของการจัดการโปรเซสขึ้นมา เรื่องราวของโปรเซสในสารพัน UNIX ตอนนี้เป็นการกล่าวถึงโปรเซสอย่างคร่าว ๆ พอให้มองภาพการทำงานของระบบยูนิกซ์ ถ้าผู้อ่านสนใจในรายละเอียดสามารถหาอ่านต่อได้จากหนังสือ Operating System ทั่วไป สำหรับตอนนี้ขอจบเพียงเท่านี้ค่อยพบกันใหม่นะครับ